Serielle Schnittstelle

Peter Winkler

Bestimmt haben Sie schon einmal davon gehört, daß Computer an den unterschiedlichsten Orten der Welt miteinander kommunizieren und Daten austauschen können. Man nutzt dabei zum Beispiel das Telefonnetz, welches zwar nur verhältnismäßig geringe Übertragungsraten zuläßt, aber dafür von beinahe überall auf der Erde Zugriff gewährt.

Damit das Telefonnetz genutzt werden kann, muß ein Modem eingesetzt werden, welches die Daten am einem Ende der Leitung in ein akustisches Signal übersetzt und am anderen Ende wieder in digitale Daten zurück konvertiert.

Die Daten werden dabei seriell übertragen, daß heißt: ein Bit nach dem anderen. Es muß dabei aber ein Protokoll eingehalten werden, damit sich die Computer und die Programme verstehen und damit Fehler in der Übertragung aufgedeckt und korrigiert werden können.

Wir wollen uns hier die serielle Übertragung mittels Nullmodemkabel näher ansehen.

Bei einem Nullmodemkabel können die Daten direkt zwischen zwei Computern übertragen werden. Das Nullmodemkabel ist lediglich ein 9– bzw. 25-poliges Kabel, bei dem die Signale paarweise ausgekreuzt sind.

Der Zugriff auf die serielle Schnittstelle erfolgt entweder direkt über die Port–Adressen der gewünschten Schnittstelle oder durch BIOS-Funktionen.

Um die serielle Schnittstelle zu programmieren, muß man zuerst feststellen, wieviele serielle Schnittstellen überhaupt auf dem Computer installiert sind, und an welchen Port–Adressen sich diese befinden.

Dieses Problem kann man mit Hilfe der BIOS-Variablen lösen. Diese Variablen befinden sich in dem Segment 0040h (siehe XT/AT-Booklet in diesem Heft).

Offset	Beschreibung
0000h	Word - Port Adresse von COM1 (1. Serielle Schnittstelle)
0002h	Word - Port Adresse von COM2 (2. Serielle Schnittstelle)
0004h	Word - Port Adresse von COM3 (3. Serielle Schnittstelle)
0006h	Word - Port Adresse von COM4 (4. Serielle Schnittstelle)

Gibt es zum Beispiel nur 2 serielle Schnittstellen, so sind an 0040:0000h und 0040:0002h die beiden Port Adressen enthalten und an den beiden anderen Adressen stehen Nullen.

Beispielfunktion, die die Portadressen ausließt.

```
int PortAnz=0;          /* Anzahl der Ports  */
unsigned int Ports[4]; /* Port Adressen     */
/*********************************************/
// void Init(void)
// Stellt fest, wieviele serielle Schnittstellen an
welchen Ports vorhanden
// sind und schreibt die Ergebnisse in globale
Variablen
/*********************************************/
void Init(void)
```

```
{
unsigned int far* ptr:  // Pointer auf
                        //die BIOS Variablen
int i:                  // Zähler

// Pointer initialisieren
ptr=(unsigned int far *)MK FP(0x0040,0):
for(i=0:i:i++)          // 4 Adressen durchlaufen
{
  Ports[i]=*(ptr+i):    // Port Adresse auslesen
  if(Ports[i]==0)break: // Überprüfen,
                        // ob Schnittstelle
  else PortAnz++:       // vorhanden ist
}
return:
}
```

Im Laufe der Zeit haben sich verschiedene Übertragungsgeschwindigkeiten und Protokolle entwickelt. Um sicherzustellen, daß beide PCs die selben Einstellungen verwenden, muß man daher zuerst beide Schnittstellen der Übertragungsstrecke je nach Wunsch konfigurieren.

Auch hier hilft uns das BIOS, das uns unter einem Interrupt eine Initialisierungsroutine bietet

Interrupt 14, Funktion 00h

Initialisierung einer seriellen Schnittstelle, Einstellung der Baud - Rate, Parität und Stop - Bits

Eingabe

```
AH      =     00h
DX      =     Nummer der seriellen
              Schnittstelle,
              beginnend bei 0
AL      =     Konfigurationsparameter
Bit 0-1       Datenlänge
                10(b)     =7 Bits
                11(b)     =8 Bits
Bit 2         Anzahl der Stop - Bits
                0(b)=1
                1(b)=1,5 bzw. 2
Bit 3-4         Paritätsprüfung
                00(b)     =keine
                01(b)     =ungerade
                11(b)     =gerade
Bit 5-7         Baud - Rate
                000(b)     =110 Baud
                001(b)     =150 Baud
                010(b)     =300 Baud
                011(b)     =600 Baud
                100(b)     =1200 Baud
                101(b)     =2400 Baud
                110(b)     =4800 Baud
                111(b)     =9600 Baud
```

Eingabe

```
AH      =     Status
        Bit 0:    Daten stehen bereit
        Bit 1:    Überlauf (nicht
bereit zum Schreiben)
        Bit 2-7:  Fehlerspezifikationen
AL      =     Modemstatus
```

Beispielfunktion, die eine serielle Schnittstelle mit dem Konfigurationsparameter – Byte initialisiert.

```
/*****************************************/
//void InitBaud(int COM, unsigned char b)
// Initialisiert COM mit den Parametern b
// Die Nummer der Schnittstelle beginnt bei 0
/*****************************************/
void InitBaud(int COM, unsigned char b)
{
asm {
mov AH,0x0   // Funktion 0
mov DX,COM   // Schnittstelle, beginnend bei 0
mov AL,b     // Konfigurations - Bitfeld
int 0x14     // Interrupt aufrufen
}
return;
}
```

Um Daten zu senden bzw. Daten zu empfangen, muß man zuerst überprüfen, ob die Schnittstelle überhaupt zum Senden bereit ist bzw. ob Daten überhaupt anstehen.

Man könnte auch hier eine BIOS-Funktion verwenden, aber wir werden hier direkt mit den Ports arbeiten.

Als erstes möchte ich zwei Funktionen vorstellen, die überprüfen, ob Daten anstehen bzw. ob die Schnittstelle bereit ist, Daten zu senden

Dazu wird jeweils ein Status–Bit überprüft, auf das mit der Port Adresse+5 zugegriffen wird.

Portadresse+5

Bit	Beschreibung
0	Daten stehen an
6	Bereit zum senden

```
/*************************************************/
//int ReadReady(int COM)
//Prüft, ob ein Datenbyte anliegt
//liefert 1 zurück, wenn ein Byte anliegt
//liefert sonst 0
/*************************************************/
int ReadReady(int COM)
{
  unsigned char b;
  b=inportb(Ports[COM]+5);
  return (b&1);
}
/*************************************************/
//int WriteReady(int COM)
//prüft, ob ein Byte seriell gesendet werden kann
//liefert 1 zurück, wenn ja
//liefert sonst 0
/*************************************************/
int WriteReady(int COM)
{
  unsigned char b;
  b=inportb(sPorts[COM]+5);
  return (b&64);
}
```

Jetzt wird's ernst!!

Um ein Byte zu senden, muß man nur überprüfen, ob die Schnittstelle bereit ist und falls das zutrifft, einfach das zu sendende Byte auf die Port Adresse schreiben.

```
/*************************************************/
//unsigned char serWrite(unsigned char b,int COM)
//sendet ein Byte b
//liefert 0 zurück, wenn Byte b gesendet wurde
//liefert sonst 0xFF
/*************************************************/
unsigned char serWrite(unsigned char b,int COM)
{
  if(WriteReady(COM))
  {
    outportb(sPorts[COM],b);
    return 0;
  }
  return 0xFF;
}
```

Das Gleiche gilt für das Lesen von Daten: überprüfen, ob ein Byte ansteht und wenn ja, einfach ein Byte von der Portadresse lesen.

```
/*************************************************/
//unsigned char Read(void)
//liest ein Byte von der seriellen Schnittstelle
//und liefet dieses zurück
//im Fehlerfall wird 0xFF zurückgeliefert
/*************************************************/
unsigned char Read(void)
{
  if(C sReadReady())
    return inportb(sPorts[sAktPort]);
  return 0xFF;
}
```

Mit diesen Funktion steht ihnen das Grundgerüst für jedes Programm, das die serielle Schnittstelle nutzen soll, zur Verfügung! Ich wünsche gutes Gelingen.